Segmentation of Printed Arabic Text

Adnan Amin

School of Computer Science and Engineering
University of New South Wales
Sydney 2052, NSW, Australia

Abstract: This paper proposes a new technique to segment Printed Arabic words into characters using Explicit Segmentation. The technique can be divided into several steps: (1) Digitization and preprocessing (2) binary tree construction; and (3) Segmentation. The advantage of this technique is that its execution does depend on either the font or size of character.

Keywords: Segmentation, Construction binary tree, Parallel Thinning, Smoothing, Arabic Characters, Pattern Recognition.

1. Introduction

Machine recognition of both printed and handwritten characters, has been intensively and extensively researched by scientists in different countries around the world. Characters written in many different languages have been recognized, e.g. from numerals to alphanumeric, and from Roman languages to Oriental languages. However, the infinite varieties of qualities and character shapes produced by handwriting and by modern printing machines have posed a major challenge to researchers in the field of pattern recognition. But the urgent need is there, and it has become increasingly evident that satisfactory solutions to this problem would be very useful in applications where it is necessary to process large volumes of printed and handwritten data, e.g. in automatic sorting of mail and cheques, payment slips, signature verification, and machine recognition and analysis of engineering drawings.

Many papers have been concerned with Latin, Chinese and Japanese characters, However, although almost a third of a billion people worldwide, in several different languages, use Arabic characters for writing, little research progress, in both on-line and off-line has been achieved towards the automatic recognition of Arabic characters. This is a result of the lack of adequate support in terms of funding, and other utilities such as Arabic text database, dictionaries, etc.

Two techniques have been applied for segmenting machine printed and handwritten Arabic words into individual characters: implicit and explicit segmentations.

(i) Implicit segmentation (straight segmentation): In this technique, words are segmented directly into letters. This type of segmentation is usually designed with rules that attempt to identify all the character's segmentation points [1, 2].

(ii) Explicit segmentation: In this case, words are externally segmented into pseudo-letters which are then recognized individually. This approach is usually more expensive due to the increased complexity of finding optimum word hypotheses [3-5].

This paper describes the design and implement of a new technique to segment an Arabic word into characters using explicit segmentation. The technique can be divide into three steps: First, is digitization and Preprocessing step in which the original image is transformed into binary image utilizing a scanner 300 dpi with some cleaning. Then the binary image is thinned using one of the existing thinning algorithms. Second, the skeleton of the image is traced from right to left and a binary tree is constructed. Finally, a segmentation algorithm is used to segment the binary tree into subtrees such that each subtree describes a character in the image.

2. General Characteristics of the Arabic Writing System

A comparison of the various characteristics of Arabic, Latin, Hebrew and Hindi scripts are outlined in Table 1. Arabic, like Hebrew, is written from right to left. Arabic text (machine printed or handwritten) is cursive in general and Arabic letters are normally connected on the base line. This feature of connectivity will be shown to be important in the segmentation process. Some machine printed and handwritten texts are not cursive, but most Arabic texts are, and thus it is not surprising that the recognition rate of Arabic characters is lower than that of disconnected characters such as printed English.

Table 1. Comparison of various scripts.

Characteristics	Arabic	Latin	Hebrew	Hindi
Justification	R-to-L	L-to-R	R-to-L	L-to-R
Cursive	Yes	No	No	Yes
Diacritics	Yes	No	No	Yes
Number of vowels	2	5	11	–
Letters shapes	1 – 4	2	1	1
Number of letters	28	26	22	40
Complementary characters	3	–	–	–

Arabic writing is similar to English in that it uses letters (which consist of 28 basic letters), numerals, punctuation marks, as well as spaces and special symbols. It differs from English, however, in its representation of vowels since Arabic utilizes various diacritical markings. The presence and absence of vowel diacritics indicates different meanings in what would otherwise be the same word. For example, □□□□□ is the Arabic word for both “school” and “teacher”. If the word is isolated, diacritics are essential to distinguish between the two possible meanings. If it occurs in a sentence, contextual information inherent in the sentence can be used to infer the appropriate meaning.

The Arabic alphabet is represented numerically by a standard communication interchange code approved by the Arab Standard and Metrology Organization (ASMO). Similar to the American Standard Code for Information Interchange (ASCII), each character in the ASMO code is represented by one byte. An English letter has two possible shapes, capital and small. The ASCII code provides separate representations for both of these shapes, whereas an Arabic letter has only one representation in the ASMO table. This is not to say, however, that the Arabic letter has only one shape. On the contrary, an Arabic letter might have up to four different shapes, depending on its relative position in the text. For instance, the letter (□ A'in) has four different shapes: at the beginning of the word (preceded by a space), in the middle of the word (no space around it), at the end of the word (followed by a space), and in isolation (preceded by an unconnected letter and followed by a space). These four possibilities are exemplified in Fig.1.

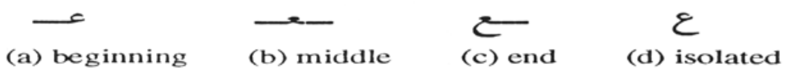


Fig. 1. Different shapes of the Arabic letter “ • A’in “

In addition, different Arabic characters may have exactly the same shape, and are distinguished from each other only by the addition of a complementary character (see Appendix). Complementary characters are positioned differently, for instance, above, below or within the confines of the character. Figure 2 depicts two sets of characters, the first set having five characters and the other set three characters. Clearly, each set contains characters which differ only by the position and/or the number of dots associated with it. It is worth noting that any erosion or deletion of these complementary characters results in a misrepresentation of the character. Hence, any thinning algorithm needs to efficiently deal with these dots so as not to change the identity of the character.

Arabic writing is cursive and is such that words are separated by spaces. However, a word can be divided into smaller units called subwords (see Appendix). Some Arabic characters are not connectable with the succeeding character. Therefore, if one of these characters exists in a word, it divides that word into two subwords. These characters appear only at the tail of a subword, and the succeeding character forms the head of the next subword. Figure 3 shows three Arabic words with one, two, and three subwords. The first word consists of one subword which has nine letters; the second has two subwords with three and one letter, respectively. The last word contains three subwords, each consisting of only one letter.

Fig. 2. Arabic characters differing only with regard to the position and number of associated dots.

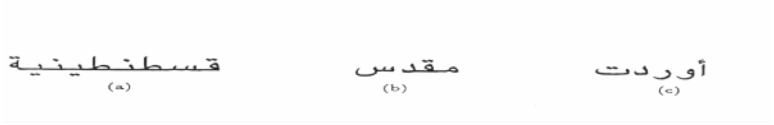


Fig. 3..Arabic words with constituent subwords.

Arabic writing is similar to Latin in that it contains many fonts and writing styles. The letters are overlaid in some of these fonts and styles. As a result, word segmentation using the baseline, which is the line on which all Arabic characters are connected, is not possible. Furthermore, characters of the same font have different sizes (i.e. characters may have different widths even though the two characters have the same font and point size). Hence, word segmentation based on a fixed size width cannot be applied to Arabic.

3. Digitization and Preprocessing

3. 1. Digitization

The first phase in our character recognition system is digitization. Documents to be processed are first scanned and digitized. A 300 dpi scanner is used to digitize the image. This generates a TIFF file which is then converted to 1-bit plane PBM file. The PBM format contains a small header which incorporates a file stamp followed by the dimensions of the image in pixels. The remainder of the file contains the image data.

3.2. Pre-thinning and Thinning

This step aims to reduce the noise due to the binarization process. The pre-thinning algorithm used in this paper is as follows:

Input. A digitized image *I* in PBM format.

Output. A pre-thinned image *I'*, also in PBM format.

```
begin
1. For each pixel $P$ in image $I$, let $P0$ to $P7$ be its 8 neighbors, starting from the east
   neighbor and counted in an anti-clockwise fashion.
2. Let $B(P) = P0 + P2 + P4 + P6$ . Let $P'$ be the corresponding pixel of $P$ in the
   pre-thinned image I'.
```

```
3.   If B (P) < 2 then set P' to white
     Else If  B(P) > 2 then set P' to black
     Else set P' to the value of P;
end
```

3. 3. Thinning

The thinning of elongated objects is a fundamental preprocessing operation in image analysis, defined as the process of reducing the width of a line-like object from several pixels to a single pixel. The resultant image is called "the skeleton".

This paper adopts Jang and Chin's one pass parallel thinning algorithm [6] because it gives skeletons with fewer spurious branches. Figure 4 illustrates an original scanned image and the resulting skeleton after applying the thinning.

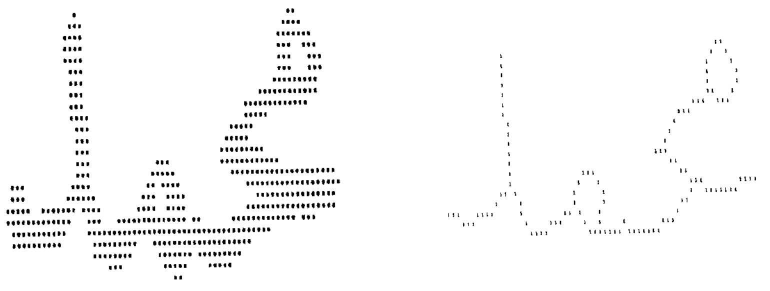

(a) (b)

Fig. 4. An example for an Arabic word before and after thinning.

After the thinning of the binary image, the structure of the image can be recorded by tracing the skeleton. However, the thinning process might alter the shape of the character, which in turn makes difficult to be recognized. Some of the common problems encountered during the thinning process include the elimination of vertical notches in some characters and elimination or erosion of complimentary characters. These modification make the recognition of the thinned image a difficult task even for nature human visual processing.

4. Thinned Image Tracing

The objective of this step in the process is to build a binary tree with all the information describing the structure of the image. The technique used involves tracing the thinned image using 3 x 3 window, and recording the structure of the traced parts. The image structure is written using some primitives such as lines, loop, and double loops etc.. This approach has a number of advantages:

(a). *Generality*: The technique can be applied to any font or size of Arabic text. In addition, it can be applied to hand printed text. Furthermore, the technique can be used in processing other language data such as Latin or Chinese.

(b). *Simple Segmentation:* The technique will simplify the segmentation process. The examination of the subword structure from right to left will identify potential points for segmentation. Some of these points are false segmentation points (i.e. they are within one character). However, all of the segmentation points (i.e. points between two consecutive characters) are part of these potential points.

The binary tree consists of several nodes. Each node describes the shape of the corresponding part of the subword. The structure of the node in the binary tree is show in Figure 5:

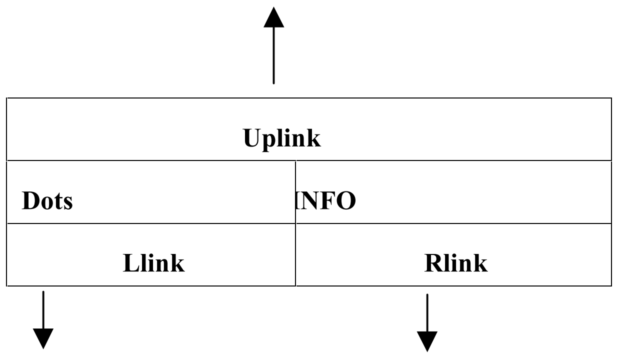


Fig. 5. The node of the binary tree.

Where Uplink is pointing to the parent node, the Rlink is pointed to the right son of that node, and Llink is pointed to the left son of that node. The Dots field signals the existence of complimentary characters in this part of the subword. The INFO field contains the primitive description of the corresponding part in the subword.

The following points are taken into consideration when building the binary tree:

1. The primitives used in the INFO field of the subtree node are the eight Freeman codes [7], 0 through 7; the Loop (L); and Double loop (LL).
2. The INFO field contains a description of the structure of the binary image which corresponds to a part of the subword. This description will terminates if a junction point is reached (the junction point is described as the point at which tracing can follow more than one possible path).
3. The L (Loop) and LL (Double loop) primitives are identified if the junction points force a looping connection within the binary tree. In other words, every time there is a pointer from a node in the ith level of the tree to another node not in the I + 1st level of the tree, it is regarded as either an L or LL primitive.
4. The tracing of the complimentary characters takes place after the termination of subword tracing. The complimentary characters, if they exust in the subword, can exist in any one of the following cases:

One. One dot above the subword baseline.
Two. One dot below the subword baseline.
Three. One dot within the subword baseline.
Four. Two dots above the subword baseline.
Five. Two dots below the subword baseline.
Six. Three dots above the subword baseline.
Seven. Zig-zag shape above the subword baseline.
Eight. Zig-zag shape within the subword baseline.
Nine. Vertical bar on some of the characters, in the case that the bar is separated from the main body of the character.

The image of the complimentary character can be identified as an image which fits into a small window. The size of the window is a parameter of the font point size. In the experiments performed, the window used was 45 pixels in perimeter. Certain features of the image within the window such as the length, width and density of the image, can be used to identify the type of complimentary character.

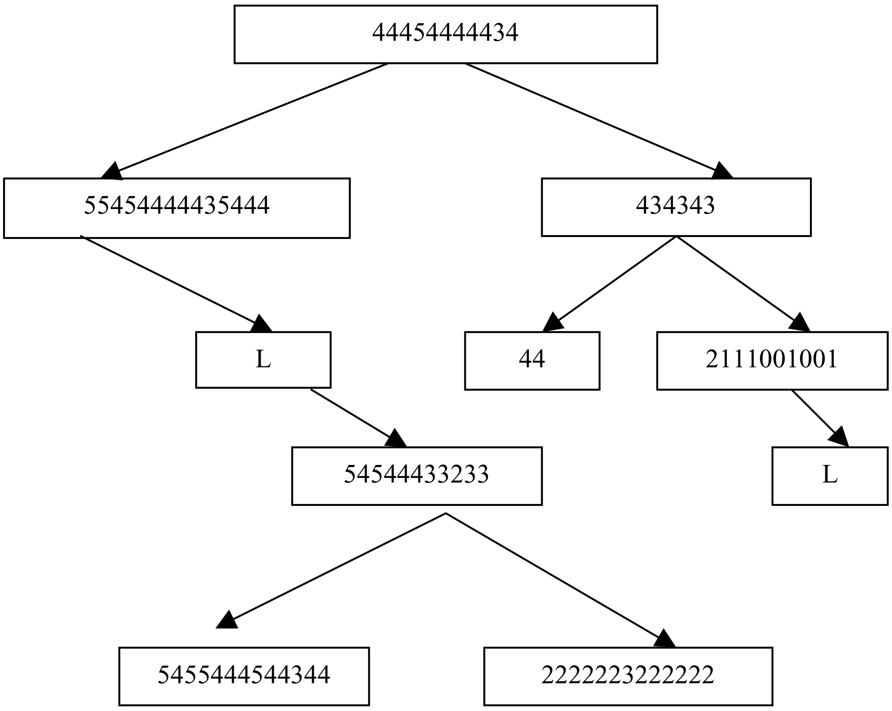


Fig. 6. Binary tree of the image in Figure 4(b)

The binary tree of the image of Figure 4(b) is depicted in Figure 6 above**.** After the construction of the binary tree takes place according to the above rules, the smoothing of the tree is performed. The smoothing of the binary tree is designed to:

(a). Minimize the number of the node in the tree.

(b). Minimize the Freeman code string in the INFO field of the nodes, and

(c). Eliminate or minimize any noise in the thinned image.

The smoothing phase of the binary tree can be summarized as follows:

(a). *Eliminated the empty nodes:* All the nodes which have an empty INFO field are eliminated. The only exception is the root node which can retain an empty INFO field. An empty INFO field is defined as the field with the null string or one Freeman code character.

(b): *Smooth the Freeman Code*: In this study, we use simple but effective smoothing algorithm, illustrated in Figure 7 for direction 0 only. The patterns and codes are rotated for other directions. Any pattern in the first column is replaced by the pattern in the second column. Thus, the string 070010 is replaced by 000000.

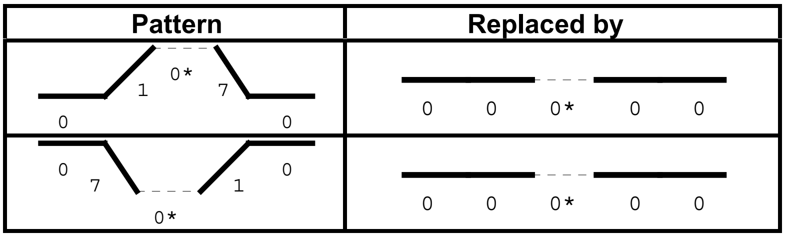


Fig. 7. Smoothing algorithm.

(c). *Compact the string in the INFO field:* Note that the string of Freeman code can be compacted by replacing it by a shorter string without any lose in information. The shorter form is the Freeman code character and its length. Therefore, if the INFO field in a node contains the following string:

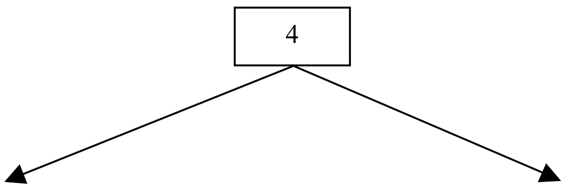
2222 4444444444 6666666

it will be replaced by

$2_4 \quad 4_{10} \quad 6_7$

The length of the Freeman code can be used later in the recognition process to remove any ambiguities if necessary.

The smoothed binary tree of the image in Figure 4 (b) is depicted in Figure 8.



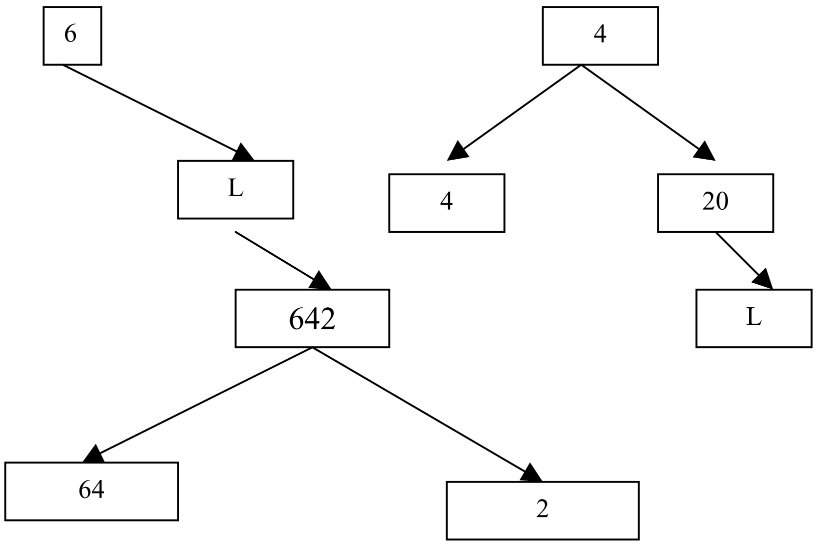


Fig. 8. The smoothed binary tree of the image in Figure 4 (b).

5. Segmentation of the Subword

Segmentation is defined as the process of dividing a subword into characters. This phase is a necessary and crucial step in recognizing printed Arabic text. To segment an Arabic subword into characters, the fundamental property of connectivity is decomposed. Segmentation techniques can be classified

There are two major problems with the traditional segmentation method which depends on the baseline:

(i) Overlapping of adjacent Arabic characters occurs naturally, see Fig. 9 (a). Hence, no baseline exists. This phenomenon is common in both typed and handwritten Arabic text.

(ii) The connection between two characters is often short. Therefore, placing the segmentation points is a difficult task. In many cases, the potential segmentation points will be placed within a character rather than between characters.

The word in Fig. 9 (a) was segmented utilizing a baseline technique. Figure 9 (b) shows the proper segmentation and the result of the new segmentation method is shown in Fig. 9 (c).

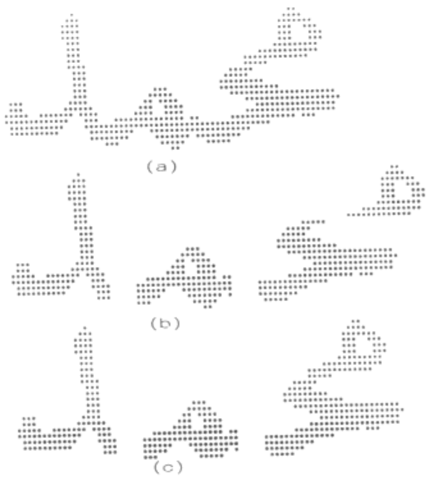


Fig. 9. Example of an Arabic word • • • • and different techniques of the segmentation.

The basic idea in the segmentation process, in this case, can be summarized in the following points:

Neun. The binary tree can be traversed in an order such that first node in the traversal process contains the beginning of the first character. However, this node is not necessarily the root node.

ii. A character description is then spread in one or more of the tree node INFO field. Nevertheless, these nodes are placed such that their traversal is undertaken one after the other depending on the shape of the character. Hence, the node which processes the beginning of a character is traversed first while the node which has the end of the character is traversed last.

iii. Some of the characters which are non-connectable from left may be contained in one node, in addition to the end of the proceeding character. In such a case, the node must be the last node traversed in the subword.

iv. A binary tree with one node indicates that the subword corresponding to that tree contains one or two characters.

During the traversal process, the subword is segmented into characters. The segmentation process can be viewed as the way to identified the beginning of the next character or realize the end of the current character. Whenever the start of a character is identified, the character "@" is concatenated to the string in the INFO field of the node. In a similar manner the character "*" is concatenated to the string in the INFO field of the node that contains the end of a character.

The process produces several subtrees each containing the Freeman code description of a character in the subword. Figure 11 depicts the tree corresponding to the subword shown in Figure 5(b). the word contains four characters. The four characters of the sunword are describes as:

1st character : in the nodes 1, 2.
2nd character: in the nodes 3, 4, 5, 6.
3rd character: in the nodes 7, 8.
4th character: in the nodes 9, 10.

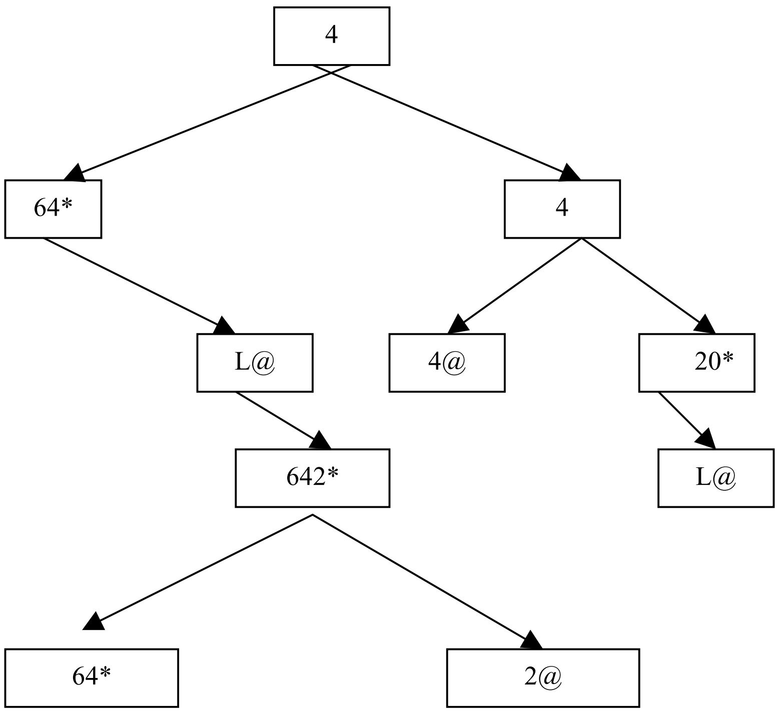


Fig. 10. Segmentation of the Binary tree shown in Figure 9.

6. Conclusion

This paper presents a new algorithm to segment Arabic word into characters. The algorithm can be applied to any font and it permits the overlay of characters. The algorithm was tested by many different fonts and size for the Arabic alphabet which ranged from excellent print to poor quality and the results was satisfactory.

The thinning of the binary image was used in order to produce a correct representation of the structure of the image by tracing the skeleton. However, due to the erosion experienced in the image, some of the characters were not segmented properly.

This is still an open research area and this is because of the segmentation problem, which is in fact similar to the segmentation of cursive script in many languages, and because of the complexity of Arabic characters.

References

1. A. Amin and G. Masini, Machine recognition of muti-fonts printed Arabic texts, *8th International Conf. on Pattern Recognition,* Paris, 392–395, 1986.
2. A. Amin, Arabic Character Recognition. Handbook of Character Recognition and Document Image Analysis edited by H Bunke and P S P Wang, May 1997.
3. H. Almuallim and S. Yamaguchi, A method of recognition of Arabic cursive handwriting, *IEEE, Trans. Pattern Anal. and Machine Intell.* PAMI-9, 715–722, 1987.
4. V. Margner, SARAT- A system for the recognition of Arabic printed text, *11th Int. Conf. on Pattern Recognition*, 1992, 561–564.
5. B. Al-Badr and R. Haralick, Segmentation-free word recognition with application to Arabic, *3rd Int. Conf. on Document Analysis and Recognition*, Montreal, 1995, 355–359.
6. B.K. Jang and R.T. Chin, One-pass parallel thinning: analysis, properties, and quantitative evaluation, *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-14, 1129-1140 (1992).
7. H. Freeman, On the encoding of arbitrary geometric configuration, *IEEE. Trans. Electronic Comp. EC-10 (1968)* 260–268.